Funnelling: A New Ensemble Method for Heterogeneous Transfer Learning and its Application to Polylingual Text Classification

ANDREA ESULI, Consiglio Nazionale delle Ricerche
ALEJANDRO MOREO, Consiglio Nazionale delle Ricerche
FABRIZIO SEBASTIANI, Consiglio Nazionale delle Ricerche

Polylingual Text Classification (PLC) consists of automatically classifying, according to a common set C of classes, documents each written in one of a set of languages $\mathcal{L}$, and doing so more accurately than when "naïvely" classifying each document via its corresponding language-specific classifier. In order to obtain an increase in the classification accuracy for a given language, the system thus needs to also leverage the training examples written in the other languages. We tackle "multilabel" PLC via *funnelling*, a new ensemble learning method that we propose here. Funnelling consists of generating a two-tier classification system where all documents, irrespectively of language, are classified by the same (2nd-tier) classifier. For this classifier all documents are represented in a common, language-independent feature space consisting of the posterior probabilities generated by 1st-tier, language-dependent classifiers. This allows the classification of all test documents, of any language, to benefit from the information present in all training documents, of any language. We present substantial experiments, run on publicly available polylingual text collections, in which funnelling is shown to significantly outperform a number of state-of-the-art baselines. All code and datasets (in vector form) are made publicly available.

ACM Reference format:
Andrea Esuli, Alejandro Moreo, and Fabrizio Sebastiani. 2016. Funnelling: A New Ensemble Method for Heterogeneous Transfer Learning and its Application to Polylingual Text Classification. 1, 1, Article 1 (January 2016), 28 pages.
DOI: 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

In *Multilingual Text Classification* each document d is written in one of a finite set $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$ of languages, and the unlabelled documents need to be classified according to a *classification scheme* $C = \{c_1, ..., c_{|C|}\}$ which is the same for all $\lambda_i \in \mathcal{L}$.

Multilingual text classification has two main subtasks, *Cross-Lingual Text Classification* (CLC) and *Polylingual Text Classification* (PLC). CLC is characterized by the fact that training examples exist only for the languages belonging to a subset $\mathcal{L}^s \subset \mathcal{L}$ (the so-called *source* languages) while no training example exists for the languages belonging to subset $\mathcal{L}^t = \mathcal{L}/\mathcal{L}^s$ (the so-called *target* languages). CLC techniques thus attempt to leverage the training examples for the languages in $\mathcal{L}^s$ in order to classify documents written in languages belonging to $\mathcal{L}^t$. In PLC, instead, we assume that there is a training set Tr_i of documents for each language $\lambda_i \in \mathcal{L}$, so that a (monolingual) classifier h_i can in principle be generated for each language λ_i. The task of PLC is thus to generate

The order in which the authors are listed is purely alphabetical; each author has given an equally important contribution to this work.

DOI: 10.1145/nnnnnnn.nnnnnnn

"enhanced" classifiers h_i^+ (i.e., classifiers whose accuracy is higher than the corresponding classifiers h_i) by leveraging the training examples for the other languages $\mathcal{L}/\{\lambda_i\}$.

Both CLC and PLC are thus instances of *transfer learning* [31, 45], i.e., are tasks in which we attempt to reuse information about a problem in a source domain, for solving the same problem in a different, target domain. More specifically, CLC and PLC are instances of *heterogeneous* transfer learning [8], i.e., are tasks in which transfer learning is performed across domains that are characterized by different feature spaces. Techniques developed for either CLC or PLC are especially useful when we need to perform text classification for under-resourced languages, i.e., languages for which only a small number (if at all) of training documents are available; in these cases, CLC or PLC techniques allow leveraging what is available for the better-resourced languages (e.g., English).

In this paper we focus on *polylingual* text classification. More specifically, we focus on general *multilabel* PLC, i.e., the PLC case in which the number of classes to which a document d belongs ranges in $\{0, ..., |C|\}$; note that multilabel PLC subsumes binary classification (which corresponds to multilabel PLC with $|C| = 1$). We propose a new, learner-independent approach for multilabel PLC that relies on *funnelling*, a 2-tier method for training classifier ensembles for heterogeneous data (i.e., data that lie in different feature spaces), which is being proposed here for the first time. In our approach a test document d_u written in language λ_i is classified by h_i^1, one among $|\mathcal{L}|$ language-specific multilabel *base classifiers*, and the output of this classifier (in the form of a vector of $|C|$ posterior probabilities $\Pr(c|d_u)$) is input to a multilabel *meta-classifier* which generates the final prediction for d_u using the latter vector as d_u's representation.

The base classifiers can actually be seen as mapping $|\mathcal{L}|$ different language-dependent feature spaces ϕ_i^1 (e.g., consisting of terms or other content features) into a common, language-independent feature space ϕ^2 (consisting of posterior probabilities). In other words, documents written in different languages, that in the 1st tier lie in different feature spaces, in the 2nd tier are "funnelled" into a single feature space. One advantage of this fact is that, as will become clear in Section 3, *all* training examples (irrespectively of language) contribute to training the meta-classifier. As a result, the classification of unlabelled documents written in *any* of the languages in $\mathcal{L}$ benefits from *all* the training examples, written in any language of $\mathcal{L}$, and thus delivers better results. Another advantage of this approach to PLC is its complete generality, since funnelling does not require the availability of multilingual dictionaries, machine translation services, or external corpora (either parallel or comparable).

This paper is structured as follows. After some discussion of related work (Section 2), in Section 3 we describe our approach to multilabel PLC in detail; in particular, in Section 3.2 we take a critical look at funnelling and at its relationships with *stacked generalization* [48], and we discuss what exactly one attempts to learn via funnelling. In Sections 4 and 5 we turn to describing the substantive experimentation to which we subject our approach; in particular, we describe experiments in multilabel PLC settings (Section 5.1), in monolingual settings and in binary settings (Section 5.2), and in settings that aim to show how funnelling may help classification for under-resourced languages (Sections 5.3 and 5.4). Section 6 takes a detour from PLC and investigates whether funnelling can also be used for performing CLC. Section 7 concludes, pointing at possible avenues for future work.

2 RELATED WORK

Initial work on PLC [2, 16] relied on standard bag-of-words representations, and investigated different preprocessing techniques with simple strategies for classification based on language-specific feature spaces (giving rise to one classifier for each language) or a single juxtaposed

feature space (giving rise to one single classifier for the entire set of languages). Since then, more sophisticated *distributional semantic models* (DSMs), such as *Cross-Lingual Latent Semantic Analysis* (CLLSA – [10]) and *Polylingual Latent Dirichlet Allocation* (PLDA – [27]), have been extensively investigated. However, the improvement in accuracy brought about by models based on these latent representations comes at a cost, since the availability of external parallel corpora (i.e., additional to the one used for training and testing purposes) is typically required.

In the absence of external parallel data, one polylingual DSM which has recently proved worthy (and that we use as a baseline in our experiments) is *Lightweight Random Indexing* (LRI – [29]), the polylingual extension of the *Random Indexing* (RI) method [36]. RI is a context-counting model belonging to the family of random projection methods, and is considered a cheaper approximation of LSA [35]. LRI is designed so that the orthogonality of the projection base is maximized, which allows to preserve sparsity and maximize the contribution of the information conveyed by the features shared across languages.

Other techniques (e.g., [13]) rely, in order to solve the multilingual classification problem, on the availability of external multilingual knowledge resources, such as dictionaries or thesauri. One of the best-known such approaches (which we will also use as a baseline in our experiments) is *Cross-Lingual Explicit Semantic Analysis* (CLESA – [39, 40]). In the original monolingual version of this technique a document is represented by a vector of similarity values, where each such value represents the similarity between the document and a predefined reference text [15]. In CLESA, different language-specific versions of the same text are considered as reference texts, so that documents written in different languages can be effectively represented in the same feature space. In a similar vein, *Kernel Canonical Correlation Analysis* (KCCA) [20], the kernelized version of CCA [22], has also been applied to cross-lingual contexts. In essence, CCA aims at maximizing the correlations among sets of variables via linear projections onto a shared space. In its application to polylingual classification, KCCA (which we will also use as a baseline in our experiments) treats language-specific views of aligned articles as different sets of variables to correlate. The projections that maximize the correlations among language-specific aligned articles are applied to the training documents in order to create a classifier.

Another method that requires external multilingual resources (specifically: a word translation oracle) is *Cross-Lingual Structural Correspondence Learning* (CL-SCL – [33]). CL-SCL relies on solving auxiliary prediction problems, which consist in discovering hidden correlations between terms in a language. This is achieved by binary classifiers trained to predict the presence of highly discriminative terms ("pivots") given the other terms in the document. The cross-lingual aspect is addressed by imposing that pivot terms are aligned (i.e., translations of each other) across languages, which requires a word translation oracle. A stronger, more recent variant of CL-SCL (which we also compare against in our experiments) is *Distributional Correspondence Indexing* (DCI – [28]). DCI derives term representations in a vector space common to all languages where each dimension reflects its distributional correspondence (as quantified by a "distributional correspondence function") to a pivot.

Machine Translation (MT) represents an appealing tool to solve PLC, and several PLC methods are indeed based on the use of MT services [34, 47]. However, the drawback of these methods is reduced generality, since it is not always the case that quality MT tools are both (i) available for the required language combinations, and (ii) free to use.

Approaches to PLC based on deep learning focus on defining representations based on word embeddings which capture the semantic regularities in language while at the same time being aligned across languages. In order to produce aligned representations, though, deep learning approaches typically require the availability of external parallel corpora [18, 23], bi-lingual lexicons

[12], or machine translation tools [1]. Recently, Conneau et al. [7] proposed a method to align monolingual word embedding spaces (as those produced by, e.g., Word2Vec [26]) from different languages without requiring parallel data. To this aim, [7] proposed an adversarial training process in which a *generator* (in charge of mapping the source embeddings onto the target space) is trained to fool a *discriminator* from distinguishing the provenance of the embeddings, i.e., from understanding whether the embeddings it receives as input come from the (transformed) source or from the target space. After that, the mapping is refined by means of unsupervised techniques. Despite operating without parallel resources, [7] obtained state-of-the-art multilingual mappings, which they later made publicly available[1] and which we use as a further baseline in our experiments of Section 4.

3 SOLVING POLYLINGUAL TEXT CLASSIFICATION VIA FUNNELLING

We now describe funnelling and its application to multilabel PLC. Let $\mathcal{L} = \{\lambda_1, ..., \lambda_{|\mathcal{L}|}\}$ be our finite set of languages, and let $C = \{c_1, ..., c_{|C|}\}$ be our finite classification scheme. Let d indicate a generic document, d_l a labelled (training) document, and d_u an unlabelled (test) document. We assume the existence of $|\mathcal{L}|$ training sets $\{Tr_1, ..., Tr_{|\mathcal{L}|}\}$ of documents, where all documents $d_l \in Tr_i$ are written in language λ_i and are labelled according to C (i.e., the set C of classes is the same for all training sets). We do not make any assumption on the relative size and composition of the different training sets; we thus allow different training sets to consist of different numbers of documents, and we do not assume the union of the training sets to be either a "parallel" dataset (i.e., consisting of translation-equivalent versions of the same documents) or a "comparable" one (i.e., consisting of documents dealing with the same events/topics although in different languages).

The first step of the training process consists of training $|\mathcal{L}|$ independent base classifiers $h^1_1, ..., h^1_{|\mathcal{L}|}$ from the respective training sets (throughout this paper the "1" superscript will indicate the 1st tier of the architecture, which consists of the base classifiers). In order to do this, for each training document $d_l \in Tr_i$ we generate a vectorial representation $\phi^1_i(d_l)$ via bag-of-words or any other standard content-based representation model; we use all the resulting vectors to train h^1_i, and repeat the process for all the Tr_i's. Quite obviously, the different base classifiers will operate in different feature spaces (for a detailed discussion of this point see the last paragraph of Section 4.3).

We do not make any assumption concerning (i) the model used for generating the vectorial representations $\phi^1_i(d)$ and (ii) the supervised learning algorithm used to train the base classifiers; it is in principle possible to use different representation models and different supervised learning algorithms for the different languages. Actually, the only assumption we make is that each trained base classifier h^1_i returns, for each document d_u written in language λ_i and for each class c, a classification score $h^1_i(d_u, c) \in \mathbb{R}$, i.e., a numerical value representing the confidence that h^1_i has in the fact that d_u belongs to c.

The second step consists of generating, for each document $d_l \in Tr_i$ and for each training set Tr_i, a vectorial representation $\phi^2(d_l)$ that will be used for training the meta-classifier. In order to do this, for each document $d_l \in Tr_i$ we first generate a vector

$$S(d_l) = (h^1_{ix}(d_l, c_1), ..., h^1_{ix}(d_l, c_{|C|})) \tag{1}$$

of $|C|$ classification scores, one per class, via k-fold cross-validation on Tr_i. In other words, we split Tr_i into k subsets $Tr_{i1}, ..., Tr_{ik}$ of approximately equal size, train a classifier h^1_{ix} (using $\phi^1_i(d)$-style vectorial representations for the training documents) using the training data in $\bigcup_{y\in\{1,...,k\}, y\neq x} Tr_{iy}$, use this classifier in order to generate vectors $S(d_l)$ of classification scores for all $d_l \in Tr_{ix}$, and repeat the process for all $1 \leq x \leq k$. The reason why we use k-fold cross-validation is that we want

[1] https://github.com/facebookresearch/MUSE

the classification scores which vector $S(d_l)$ is composed of, to be generated by classifiers trained on data that do not contain d_l itself.

All training documents, irrespectively of the language they are written in, thus give rise to (dense) vectors $S(d_l)$ of classification scores, and these vectors are all in the same vector space. In other words, should we view a document d_l as represented by vector $S(d_l)$, all documents would be represented in the same feature space, i.e., the space of base classifier scores for classes $C = \{c_1, ..., c_{|C|}\}$. We could thus in principle use the set $\{S(d_l) \mid d_l \in \bigcup_{i=1}^{|\mathcal{L}|} Tr_i\}$ as a large unified training set for training a meta-classifier for C. This is indeed what we are going to do, but before doing this we transform all vectors $S(d_l)$ of classification scores into vectors of $|C|$ posterior probabilities

$$\begin{aligned}\phi^2(d_l) &= (\Pr(c_1|d_l), ..., \Pr(c_{|C|}|d_l)) \\ &= (f_{ix}(h^1_{ix}(d_l, c_1)), ..., f_{ix}(h^1_{ix}(d_l, c_{|C|})))\end{aligned} \tag{2}$$

where $\Pr(c_j|d_l)$ represents the probability that the originating base classifier attributes to the fact that d_l belongs to c_j, and where f_{ix} is a mapping to be discussed shortly. Note that the $\Pr(c_j|d_l)$'s are just subjective estimates generated by the classifiers, and are not probabilities in any "objective" sense (whatever this might mean).

The rationale for not using the original classification scores $h^1_{ix}(d_l, c_j)$ as features is that vectors of classification scores coming from different classifiers are not comparable with each other (see [4, §7.1.3] for a discussion), and it would thus be unsuitable to use them *together* as feature vectors in the same training set. The task of finding a function f_{ix} that maps classification scores into posterior probabilities while at the same time obtaining "well calibrated" (i.e., good) posterior probabilities, is referred to as *probability calibration*,[2] and several methods for performing it are known from the literature (see e.g., [32, 49]). We perform probability calibration independently for each of the $|\mathcal{L}|$ training sets and each of the k folds (since each of these $|\mathcal{L}| \times k$ settings yields a different classifier), thus resulting in $|\mathcal{L}| \times k$ different calibration functions $f_{11}, ..., f_{|\mathcal{L}|k}$.

The net result is that all the vectors in $\{\phi^2(d_l)|d_l \in \bigcup_{i=1}^{|\mathcal{L}|} Tr_i\}$ are now comparable, and can thus be safely used for training the meta-classifier h^2. Here we do not make any assumption concerning the learning algorithm used to train h^2, the only requirement being that it needs to accept non-binary vectorial representations as input. In particular, it is in principle possible to train our meta-classifier via a learning algorithm different from the one used to train the base classifiers.

As a final step of the learning process we perform probability calibration for the base classifiers $h^1_1, ..., h^1_{|\mathcal{L}|}$ trained in the first step, thus giving rise to additional $|\mathcal{L}|$ calibration functions $f_1, ..., f_{|\mathcal{L}|}$.

The classification process follows the steps already outlined in Section 1. An unlabelled document d_u written in language $\lambda_i \in \mathcal{L}$ is classified by its corresponding language-specific base classifier h^1_i. The resulting vector of classification scores $S(d_u)$ is mapped into a vector $\phi^2(d_u)$ of posterior probabilities by the function f_i obtained via probability calibration in the last step of the training process. Vector $\phi^2(d_u)$ is fed to classifier h^2, which generates $|C|$ binary classification decisions $h^2(d_u, c_1), ..., h^2(d_u, c_{|C|})$.

[2]Posterior probabilities $\Pr(c|d)$ are said to be *well calibrated* when $\lim_{|S|\to\infty} \frac{|\{d \in c \mid \Pr(c|d)=x\}|}{|\{d \in S \mid \Pr(c|d)=x\}|} = x$ [9]. Intuitively, this property implies that, as the size of the sample S goes to infinity, e.g., 90% of the documents $d \in S$ such that $\Pr(c|d) = 0.9$ belong to class c. Some learning algorithms (e.g., AdaBoost, SVMs) generate classifiers that return confidence scores that are not probabilities, since these scores do not range on [0,1]; in this case, a calibration phase is needed to convert these scores into well calibrated probabilities. Other learning algorithms (e.g., Naïve Bayes) generate classifiers that output probabilities that are not well calibrated; in this case too, a calibration phase is necessary in order to obtain well calibrated probabilities. Yet other learning algorithms (e.g., logistic regression) are known to generate classifiers that already return well calibrated probabilities; in these cases no separate calibration phase is necessary.

We call our method Fun(kfcv) – with kfcv standing for "k-Fold Cross-Validation" – in order to distinguish it from a variant to be discussed in Section 3.1.

3.1 Two variants of funnelling

One problem with Fun(kfcv) is that the representations $\phi^2(d_l)$ of the labelled documents used to train the meta-classifier h^2 may not match well (i.e., faithfully represent) the representations $\phi^2(d_u)$ of the unlabelled documents that will be fed to h^2, and this would contradict the basic assumption of supervised learning. In fact, (assuming for simplicity that both d_l and d_u are written in the same language λ_i) the posterior probabilities of which $\phi^2(d_u)$ consists of have been generated by classifier h_i^1, which has been trained on the entire set Tr_i, while the posterior probabilities of which $\phi^2(d_l)$ consists of, have been generated by one of the classifiers h_{ix}^1 trained during the k-fold cross-validation process, which has been trained on a *subset* of Tr_i of cardinality $\frac{k-1}{k}|Tr_i|$.

In other words, the base classifier h_i^1 that classifies the unlabelled documents has received *more* training than the base classifiers h_{ix}^1 that classified the training data; this difference may be especially substantial for low-frequency classes, where decreasing the size of the training set sometimes means depleting an already tiny set of positive training examples. As a result, the posterior probabilities $\Pr(c_j|d_u)$ for the unlabelled documents tend to be different (actually: higher-quality) than the corresponding posterior probabilities $\Pr(c_j|d_l)$ for the training documents. Because of this mismatch, the meta-classifier h^2 may perform suboptimally.

In order to minimize this mismatch one could arbitrarily increase the number k of folds, maybe even using leave-one-out validation (i.e., k-fold cross-validation with $k = |Tr_i|$). However, this solution is computationally impractical, since a high value of k implies not only a high number of training rounds, but also a high number of probability calibration rounds (since, as already observed, calibration needs to be done independently for each trained classifier), which is expensive since calibration usually entails extensive search in a space of parameters.

An alternative, radically simpler solution might consist in doing away with k-fold cross-validation. In this solution (that we will call Fun(tat), where tat stands for "Train and Test"), Equations 1 and 2 would be replaced by

$$S(d_l) = (h_i^1(d_l, c_1), ..., h_i^1(d_l, c_{|C|})) \tag{3}$$

$$\phi^2(d_l) = (\Pr(c_1|d_l), ..., \Pr(c_{|C|}|d_l)) \tag{4}$$

$$= (f_i(h_i^1(d_l, c_1)), ..., f_i(h_i^1(d_l, c_{|C|}))) \tag{5}$$

i.e., the vectors of $|C|$ scores $S(d_l)$ and the vectors $\phi^2(d_l)$ of $|C|$ posterior probabilities would be generated directly by the classifiers h_i^1 trained on the entire training set Tr_i (with the help of the calibration functions f_i discussed towards the end of the previous section). Note that Fun(tat) entails just $|\mathcal{L}|$ training and calibrations rounds, while Fun(kfcv) entails $|\mathcal{L}| \times (k + 1)$.

Fun(tat) is not exempt from problems either, and actually suffers from the *opposite* drawback with respect to Fun(kfcv). Here again, the representations $\phi^2(d_l)$ of the labelled documents used to train the meta-classifier may not match well the representations $\phi^2(d_u)$ of the unlabelled documents, for the simple reason that classifier h_i^1 classifies (in order to generate the representations $\phi^2(d_l)$ to be used for training the meta-classifier) the very same training examples d_l it has been trained on. As a result, the posterior probabilities $\Pr(c_j|d_u)$ for the unlabelled documents tend to be *lower*-quality (hence different) than the corresponding posterior probabilities $\Pr(c_j|d_l)$ for the training documents, since documents d_u have not been seen during training.

The two variants have thus opposite pros and cons; as a result, in our experiments we will test both of them, side by side. Both variants are collectively described in pseudocode form as Algorithm 1, where the **if** command of Line 4 determines which of the two variants is executed.

ALGORITHM 1: Funnelling for multilabel PLC; the **if** command of Line 4 chooses which of FUN(KFCV) and FUN(TAT) is executed.

Input : • Sets $\{Tr_1, \ldots, Tr_{|\mathcal{L}|}\}$ of training documents written in languages $\mathcal{L} = \{\lambda_1, \ldots, \lambda_{|\mathcal{L}|}\}$, all labelled according to sets of classes $C = \{c_1, \ldots, c_{|C|}\}$;
• Sets $\{Te_1, \ldots, Te_{|\mathcal{L}|}\}$ of unlabelled documents written in languages $\mathcal{L} = \{\lambda_1, \ldots, \lambda_{|\mathcal{L}|}\}$, all to be labelled according to sets of classes $C = \{c_1, \ldots, c_{|C|}\}$;
• Flag *Variant*, with values in $\{$*FUN(KFCV)*, *FUN(TAT)*$\}$

Output : • 1st-tier language-specific classifiers $h^1_1, \ldots, h^1_{|\mathcal{L}|}$;
• 2nd-tier language-independent classifier h^2 ;
• Labels for all documents in $\{Te_1, \ldots, Te_{|\mathcal{L}|}\}$;

```
/* Training phase */
1  for λ_i ∈ L do
       /* Train 1st-tier classifiers and find a calibration function for them */
2      Train classifier h^1_i from Tr_i;
3      Compute calibration function f_i via chosen calibration method;
       /* Generate vectors of posterior probabilities for training meta-classifiers */
4      if Variant="FUN(KFCV)" then
           /* Use the Fun(kfcv) variant of the algorithm */
5          Split Tr_i into k folds {Tr_i1,...,Tr_ik};
6          for 1 ≤ x ≤ k do
7              Train classifier h^1_ix from ⋃_{y∈{1,...,k}, y≠x} Tr_iy;
8              Compute calibration function f_ix via chosen calibration method;
9              for d_l ∈ Tr_ix do
                   /* Compute vector of calibrated posterior probabilities */
10                 φ^2(d_l) ← (f_ix(h^1_ix(d_l, c_1)), ..., f_ix(h^1_ix(d_l, c_|C|))) ;
11             end
12         end
13     else
           /* Use the Fun(tat) variant of the algorithm */
14         for d_l ∈ Tr_i do
               /* Compute vector of calibrated posterior probabilities */
15             φ^2(d_l) ← (f_i(h^1_i(d_l, c_1)), ..., f_i(h^1_i(d_l, c_|C|))) ;
16         end
17     end
18 end
19 Train classifier h^2 from all vectors φ^2(d_l);
   /* Classification phase */
20 for λ_i ∈ L do
21     for d_u ∈ Te_i do
           /* Compute vector of calibrated posterior probabilities */
22         φ^2(d_u) ← (f_i(h^1_i(d_u, c_1)), ..., f_i(h^1_i(d_u, c_|C|))) ;
           /* Invoke meta-classifier */
23         Compute h^2(d_u, c_1), ..., h^2(d_u, c_|C|) from φ^2(d_u).
24     end
25 end
```

3.2 What does funnelling learn, exactly?

Funnelling is reminiscent of the *stacked generalization* (a.k.a. "stacking") method for ensemble learning [48]. Let us discuss their commonalities and differences.

Common to stacking and funnelling is the presence of an ensemble of n base classifiers, typically trained on "traditional" vectorial representations, and the presence of a single meta-classifier that operates on vectors of base-classifier outputs. Common to stacking and FUN(KFCV) is also the use of k-fold cross-validation in order to generate the vectors of base-classifier outputs that are used to train the meta-classifier. (Variants of stacking in which k-fold cross-validation is not used, and thus akin to FUN(TAT), also exist [37].)

However, a key difference between the two methods is that stacking (like other ensemble methods such as bagging [5] and boosting [14]) deals with ("homogeneous") scenarios in which all training documents can in principle be represented in the same feature space and can thus concur to training the same classifier; in turn, this classifier can be used for classifying all the unlabelled documents. In stacking, the base classifiers sometimes differ in terms of the learning algorithm used to train them [37, 43], or in terms of the subsets of the training set which are used for training them [6]. In other words, in these scenarios setting up an ensemble is a choice, and not a necessity. It is instead a necessity in the ("heterogeneous") scenarios which funnelling deals with, where labelled documents of different types (in our case: languages) could otherwise *not* concur in training the same classifier (since they lie in different feature spaces), and where unlabelled documents could not (for analogous reasons) be classified by the same classifier.

The consequence is that, while in stacking *all* base classifiers classify the test document, in funnelling only one base classifier does this.[3] In turn, this means that in stacked generalization the length of the vectors on which the meta-classifier operates is $n \cdot |C|$ (with n the number of base classifiers), while it is just $|C|$ in funnelling. In stacking, n different scores (one for each base classifier) for the same (d_u, c) test pair are thus received by the meta-classifier, who then needs to combine them in order to reach a final decision. As noted in [11], stacking is indeed a method for *learning to combine* the n scores returned by a set of n base classifiers for the same (d_u, c) test pair. While in many classifier ensembles a static combination rule – e.g., weighted voting – is used to combine the outputs of the individual base classifiers, in stacking this combination rule is learned from data. By contrast, there is no combination of different outputs in funnelling, since a document is always classified by only one base classifier. Graphical depictions of the architectures of funnelling and stacking are given in Figure 1.

So, if the meta-classifier of an ensemble built via funnelling does not learn to combine different scores for the same (d_u, c) pair, what does it learn exactly?

It certainly *learns to exploit the stochastic dependencies between classes* that exist in multilabel settings [17, 30, 44], which is not possible when (as customarily done) a multilabel classification task is solved as $|C|$ independent binary classification problems. In fact, for an unlabelled document d_u the meta-classifier receives $|C|$ inputs from the base classifier which has classified d_u, and returns $|C|$ outputs, which means that the input for class c' has a potential impact on the output for class c'', for every choice of c' and c''. For instance, the fact that for d_u the posterior probability for class `Skiing` is high might bring additional evidence that d_u belongs to class `Snowboarding`; this could be the result of several training documents labelled by `Snowboarding` having, in their $\phi^2(d)$ vectors, a high value for class `Skiing`.

[3] Kuncheva [24, p. 106] observes that "It is accepted now that there are two main strategies in combining classifiers: fusion and selection. In classifier fusion, each ensemble member is supposed to have knowledge of the whole feature space. In classifier selection, each ensemble member is supposed to know well a part of the feature space and be responsible for objects in this part." Funnelling is thus an instance of the "classifier selection" strategy for creating an ensemble.

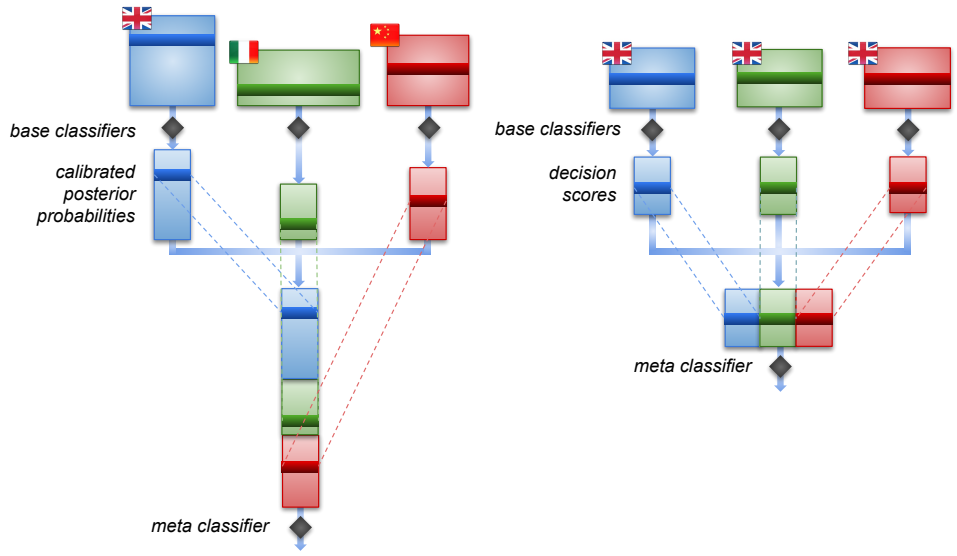

Fig. 1. Architectures of a classifier system based on funnelling (left) and of one based on stacking (right). Black diamonds represent individual classifiers, dark thin rectangles represent individual vectors, while larger coloured rectangles that contain them represent sets of vectors; national flags represent the different languages on which language-specific classifiers operate. The fact that, in funnelling, the larger coloured rectangles at the top have different widths indicates that the sets of vectors they represent lie in different feature spaces, which may have different dimensionalities (this is usually not the case in stacking); the fact that they have different heights indicates that the sets of vectors they represent may come in different sizes (this is usually not the case in stacking either); above all, the fact that they are labelled by different national flags indicates that the sets of vectors they represent lie in different feature spaces.

However, learning to exploit the stochastic dependencies between different classes is certainly not the primary motivation behind funnelling. The primary motivation is instead *learning from heterogeneous data*, i.e., data that come in n different, incomparable varieties, and that because of the differences among these varieties require n completely different feature spaces to accommodate them. When all these diverse data need to be classified, despite their diversity, according to a common classification scheme C, funnelling can be used to set up a single classifier (the meta-classifier) that handles them all. Funnelling can be seen as mapping n different, incomparable feature spaces into a common, more abstract feature space in which all differences among the original n feature spaces have been factored out. As a result, the meta-classifier can be trained from the union of the n training sets, which means that *all* training examples, irrespectively of their provenance, concur to the common goal of classifying *all* the unlabelled examples, irrespectively of the provenance of each of these.

4 EXPERIMENTAL SETTING

4.1 Datasets

We perform our experiments on two publicly available datasets, RCV1/RCV2 (a comparable corpus) and JRC-Acquis (a parallel corpus).[4]

[4]All the information required to replicate the experiments, e.g., IDs of the selected documents, assigned labels, code, etc., is made available at https://github.com/AlexMoreo/funnelling.

4.1.1 RCV1/RCV2. RCV1-v2 is a publicly available collection consisting of the 804,414 English news stories generated by Reuters from 20 Aug 1996 to 19 Aug 1997 [25]. RCV2 is instead a polylingual collection, containing over 487,000 news stories in one of thirteen languages other than English (Dutch, French, German, Chinese, Japanese, Russian, Portuguese, Spanish, LatinoAmerican Spanish, Italian, Danish, Norwegian, Swedish), and generated by Reuters in the same timeframe. The documents of both collections are classified according to the same hierarchically organized set of 103 classes. The union of RCV1-v2 and RCV2 (hereafter referred to as RCV1/RCV2) is a corpus *comparable* at topic level, as news stories are not direct translations of each other but simply discuss the same or related events in different languages. Since the corpus is not parallel, a training document for a given language does not have, in general, a counterpart in the other languages.

In our RCV1/RCV2 experiments we restrict our attention to the 9 languages (English, Italian, Spanish, French, German, Swedish, Danish, Portuguese, and Dutch) for which stop word removal and lemmatization are supported in NLTK[5]. In order to give equal treatment to all these languages, from RCV1/RCV2 we randomly select 1,000 training and 1,000 test news stories for each language (with the sole exception of Dutch, for which only 1,794 documents are available, and for which we thus select 1,000 documents for training and 794 for test); this allows us to run our experiments in controlled experimental conditions, i.e., to minimize the possibility that the effects we observe across languages are due to different amounts of training data for the different languages tested upon.[6]

Following this selection, we limit our consideration to the 73 classes (out of 103) that end up having at least one positive training example, in any of the 9 languages. As a result, the average number of classes per document is 3.21, ranging from a minimum of 1 to a maximum of 13; the number of positive examples per class ranges from a minimum of 1 to a maximum of 3,913. The average number of distinct features (i.e., word lemmas) per language is 4,176, with a total of 26,977 distinct terms across all languages, of which 10,613 appear in two or more languages.

Since the selection of 1,000 training and 1,000 test documents for each language introduces a random factor, we repeat the entire process 10 times, each time with a different random selection; all the RCV1/RCV2 results we report in this paper are thus averages across these 10 random trials.

4.1.2 JRC-Acquis. JRC-Acquis (version 3.0) is a collection of parallel legislative texts of European Union law written between the 1950s and 2006 [42]. JRC-Acquis is publicly available for research purposes, and covers 22 official European languages. The corpus is parallel and aligned at the sentence level, i.e., of each document there are 22 language-specific versions which are sentence-by-sentence translations of each other. The dataset is labelled according to the EuroVoc thesaurus, which consists of a hierarchy of more than 6,000 classes; for our experiments we select the 300 most frequent ones.

We restrict our attention to the 11 languages (the same 9 languages of RCV1/RCV2 plus Finnish and Hungarian) for which stop word removal and lemmatization are supported in NLTK (we do not consider Romanian due to incompatibilities found in the source files).

For inclusion in the training set we take all documents written in the [1950,2005] interval and randomly select, for each of them, one of the 11 language-specific versions. The rationale of this policy is to avoid the presence of translation-equivalent content in the training set; this will enable

[5] http://www.nltk.org/

[6] The above selection protocol allows us to *minimize* the effects due to the amounts of training data available for the different languages, but not to *eliminate* them. The reason is that different training examples may have different number of classes associated to them, so one example that has more of them contributes more training information than an example that has fewer of them. This is a factor that is almost impossible to eliminate from a multilabel dataset.

us to measure the contribution of training information coming from different languages in a more realistic setting.

For the test set we instead take all documents written in 2006 and retain all their 11 language-specific versions. The rationale behind this policy is to allow a perfectly fair evaluation across languages, since each of the 11 languages is thus evaluated on exactly the same content. This process results in 12,687 training documents (between 1,112 and 1,198 documents per language) and 46,662 test documents (exactly 4,242 documents per language). The average number of classes per document is 3.31, ranging from a minimum of 1 to a maximum of 18; the number of positive examples per class ranges from a minimum of 55 to a maximum of 1,155. There is an average of 9,909 distinct word lemmas per language, a total of 81,458 distinct terms across all languages, of which 27,550 appear in more than one language.

As in RCV1/RCV2, we repeat the process of selecting training data 10 times, each time with a different random selection (this means that, in each of these 10 random trials, a different language-specific version of the same document is selected); for JRC-Acquis too, all the results we report in this paper are thus averages across these 10 random trials.

4.2 Evaluation measures

As the evaluation measures for binary classification we use both the "classic" F_1 and the more recently proposed K [38]. These two functions are defined as

$$F_1 = \begin{cases} \dfrac{2TP}{2TP + FP + FN} & \text{if } TP + FP + FN > 0 \\ 1 & \text{if } TP = FP = FN = 0 \end{cases} \tag{6}$$

$$K = \begin{cases} \dfrac{TP}{TP + FN} + \dfrac{TN}{TN + FP} - 1 & \text{if } TP + FN > 0 \text{ and } TN + FP > 0 \\ 2\dfrac{TN}{TN + FP} - 1 & \text{if } TP + FN = 0 \\ 2\dfrac{TP}{TP + FN} - 1 & \text{if } TN + FP = 0 \end{cases} \tag{7}$$

where TP, FP, FN, TN, represent the numbers of true positives, false positives, false negatives, true negatives, generated by a binary classifier. F_1 ranges between 0 (worst) and 1 (best); K ranges between -1 (worst) and 1 (best), with 0 corresponding to the accuracy of the random classifier.

In order to turn F_1 and K into measures for *multilabel* classification we compute their "microaveraged" versions (indicated as F_1^{μ} and K^{μ}) and their "macroaveraged" versions (indicated as F_1^{M} and K^{M}). F_1^{μ} and K^{μ} are obtained by (i) computing the class-specific values TP_j, FP_j, FN_j, TN_j; (ii) obtaining TP as the sum of the TP_j's (same for FP, FN, TN), and then (iii) applying Equations 6 and 7. F_1^{M} and K^{M} are obtained by first computing the class-specific values of F_1 and K and then averaging them across all $c_j \in C$.

In all cases we also report the results of paired sample, two-tailed t-tests at different confidence levels ($\alpha = 0.05$ and $\alpha = 0.001$) in order to assess the statistical significance of the differences in performance as measured by the averaged results.

4.3 Representing text

We preprocess text by using the stop word removers and lemmatizers available for all our languages within the `scikit-learn` framework[7]. As the weighting criterion we use a version of the well-known *tfidf* method, expressed as

$$tfidf(f,d) = \log \#(f,d) \times \log \frac{|Tr_i|}{|d' \in Tr_i : \#(f,d') > 0|} \tag{8}$$

where $\#(f,d)$ is the raw number of occurrences of feature f in document d and λ_i is the language d is written in; weights are then normalized via cosine normalization, as

$$w(f,d) = \frac{tfidf(f,d)}{\sqrt{\sum_{f' \in F_i} tfidf(f',d)^2}} \tag{9}$$

Our feature spaces F_i resulting from the different, language-specific training sets Tr_i are non-overlapping, since (consistently with most multilingual text classification literature) we do not make any attempt to detect matches between features across different languages. Detecting such matches would be problematic, since identical surface forms do not always translate to identical meanings; e.g., while word `Madrid` as detected in a Spanish text and word `Madrid` as detected in an Italian text may have the same meaning, word `burro` as detected in a Spanish text and word `burro` as detected in an Italian text typically do not (`burro` means "donkey" in Spanish and "butter" in Italian). The main reason why we do not attempt to detect such matches is that neither funnelling (which uses different base classifiers for the different languages) nor any of the baseline systems we use (see Section 4.4) would gain any advantage even from a hypothetically perfect detection of such matches.

4.4 Baselines

We choose the following polylingual methods as the baselines against which to compare our approach (see also Section 2 for more detailed descriptions of these methods):

- NAÏVE: This method consists in classifying each test document by a monolingual classifier trained on the corresponding language-specific portion of the training set; thus, there is no contribution from the training documents written in other languages. NAÏVE is usually considered a lower bound for any PLC effort.
- LRI: *Lightweight Random Indexing* [29], a PLC method that does not use any external resource. In all experiments we set the dimensionality of the reduced space to 25,000.
- CLESA: *Cross-Lingual Explicit Semantic Analysis* [41]. Unlike LRI and Funnelling, CLESA does require external resources, in the form of a comparable corpus of reference texts. In our experiments, consistently with the CLESA literature, as the reference texts we use 5,000 Wikipedia pages randomly chosen among the ones that (i) exist for all the languages in our datasets, and (ii) contain 50 words or more in each of their language-specific versions. We use the Wikipedia Extractor tool[8] to obtain clean text versions of Wikipedia pages from a Wikipedia XML dump. The tool filters out any other information or annotation present in Wikipedia pages, such as images, tables, references, and lists.
- KCCA: *Kernel Canonical Correlation Analysis* [46]. We use the `Pyrcca` [3] package to implement a polylingual classifier based on KCCA. Since `Pyrcca` does not provide specialized

[7] http://scikit-learn.org/
[8] http://medialab.di.unipi.it/wiki/Wikipedia_Extractor

data structures for storing sparse matrices[9], the amount of memory it requires in order to allocate all the language-specific views of the term co-occurrence matrices grows rapidly. In order to keep computation times within acceptable bounds, in our experiments we thus limit the number of comparable documents (for which we use Wikipedia articles, as for CLESA) to 2000 (and not 5000, as we do for CLESA). We set the number of components to 1000 and (after optimization via k-fold cross-validation) the regularization parameter to 1 for RCV1/RCV2 and to 10 for JRC-Acquis.

- DCI: *Distributional Correspondence Indexing*, as described in [28], and adapted to the polylingual setting by using the category labels (instead of a subset of terms) as the pivots. The dimensionality of the embedding space is thus set to the number of classes. In our experiments, as the distributional correspondence function (see [28]) we adopt the linear one, since in preliminary experiments (not reported here for the sake of brevity) in which we used different such functions it proved the best one.
- PLE: *Poly-Lingual Embeddings* derives document representations based on the multilingual word embeddings (of size 300) released by Conneau et al. [7]. As proposed by the authors, documents are represented as an aggregation of the embeddings associated to the words they contain; since the word embeddings are aligned across languages, the documents end up being represented in the same vector space, irrespectively of the language they are written in. Given that we are representing documents (and not sentences as in [7]), we weigh each embedding by its *tfidf* score (instead of by its *idf* score as suggested in [7]), in order to better reflect the relevance of the term in the document (we have indeed verified *tfidf* to perform better than simple *idf* in preliminary experiments, which we do not discuss for the sake of brevity).
- PLE-LSTM: Averaging embeddings causes a loss of word-order information. Modern NLP approaches attempt to capture such information by training Recurrent Neural Networks (RNNs) via "backpropagation through time". PLE-LSTM uses a Long Short-Term Memory (LSTM) cell [21] as the recurrent unit which, by processing sequences of embeddings, produces a document embedding that is then passed through a series of feed-forward connections with non-linear activations to finally derive a vector of probabilities for each class. The embeddings are initialized in PLE-LSTM with the multilingual embeddings released by Conneau et al. [7], and are fine-tuned during training. We use 512 hidden units in the recurrent cell, and 2048 units in the next-to-last feed-forward layer. The non-linear connection between layers is the ReLU (REctifier Linear Unit), and a 0.5 dropout is applied to every layer and recurrent connections in order to prevent overfitting. We use the RMSprop optimizer [19] with default parameters to minimize the binary cross-entropy loss of the posterior probabilities with respect to the labels. We train the network through 200 epochs in RCV1/RCV2 and through 2000 epochs in JRC-Acquis, until convergence, with an early-stopping criterion that terminates the training after p epochs show no improvement on the held-out validation set (a random sample containing 20% of the training data); p is the *patience* parameter, that we set to 20 for RCV1/RCV2 and to 200 for JRC-Acquis. Note that this is the only method among all the tested ones that accounts for word-order information.
- UpperBound: This is not a real (or realistic) baseline, but a system only meant to act, as the name implies, as an idealized upper bound that all PLC methods should strive to

[9] `Pyrcca` is primarily optimized for working not on texts but on images. Still, it is the only available implementation we are aware of that allows to learn projections for more than two sets of variables.

emulate (although its performance is hard to reach in practice). In UpperBound each non-English training example is replaced by its corresponding English version, a monolingual English classifier is trained, and all the English test documents are classified. We deploy UpperBound only for the JRC-Acquis dataset (where this gives rise to a training set of 12,687 English documents), since in RCV1/RCV2 the English versions of non-English training examples are not available.

Note that, despite the fact that ours is an ensemble learning method, we do not include other such methods as baselines. The reason is that other ensemble learning methods (such as e.g., stacking, bagging, or boosting) inherently deal (as already noted in Section 3.2) with "homogeneous" settings, i.e., scenarios in which all examples lie in the same feature space. PLC is a "heterogeneous" setting, in which examples written in different languages lie in different feature spaces, and the above-mentioned methods are not equipped for dealing with these scenarios. In fact, to the best of our knowledge, ours is the first ensemble learning method in the literature that can deal with heterogeneous settings.

4.5 Learning algorithms

We have implemented our methods and all the baselines as extensions of `scikit-learn`.

As the learning algorithm we use Support Vector Machines (SVMs), in the implementation provided by `scikit-learn`. As customary in multilabel classification, each 1st-tier multilabel classifier is simply a set of independently trained binary classifiers, one for each class $c \in C$.

Note that, when training a Fun(tat) classifier, when for a certain (λ_i, c_j) pair there are no positive training examples, we generate a trivial rejector, i.e., a classifier h_i^1 that returns scores $h_i^1(d_u, c_j) = 0$ (and, as a consequence, posterior probabilities $\Pr(c_j|d_u) = 0$) for all test documents d_u written in language λ_i. In our datasets this can indeed happen since, while we remove from both datasets the classes that do not have any positive training examples, not all remaining classes have positive training examples *for every language*.

For the k-fold cross-validation needed in the Fun(kfcv) method we use $k = 10$. We should also remark that, when training a Fun(kfcv) classifier, splitting the training set Tr_i into $Tr_{i1}, ..., Tr_{ik}$ might end up in placing all the positive training examples in the same subset Tr_{ix} (this always happens when there is a single positive training example for (λ_i, c_j)), which means that we would be left with no positive training examples for training classifier h_{ix}^1. In this case, instead of generating (as in the Fun(tat) case discussed above) a classifier h_{ix}^1 that works as a trivial rejector, we train h_{ix}^1 via Fun(tat), i.e., by also using the training examples in Tr_{ix}. In preliminary experiments that we have carried out on a separate dataset, the use of this simple heuristics has brought about substantial benefits; as a result we have adopted it in all the experiments reported in this paper.[10]

We optimize the C parameter, which controls the trade-off between the training error and the margin of the SVM classifier, through a 5-fold cross-validation on the training set, via grid search on $\{10^{-1}, 10^0, \ldots, 10^4\}$; we do this optimization individually for each method and for each run. For the two funnelling methods we perform this grid search only for the meta-classifier, leaving C to its default value of 1 for the base classifiers; the main reason is that, especially in the case of Fun(kfcv) (where an expensive 10-fold cross validation is already performed in order to generate

[10] One might wonder why, in order to avoid the possibility that the union of $(k-1)$ folds contains zero positive examples of a given class, when training Fun(kfcv) we do not use *stratified* k-fold cross-validation (which consists in choosing the k folds in such a way that the class prevalences in each fold are approximately equal to the class prevalences in the entire training set). There are two reasons for this. First, using stratification would not eradicate the problem, because there are many pairs (λ_i, c_j) for which there are ≤ 1 positive examples in the entire training set. Second, stratification is convenient for binary or single-label classification, but not for multilabel classification, where a different split into k folds must be set up for each different class. For these reasons we opt for using the traditional (non-stratified) variant.

		NAÏVE	LRI	CLESA	KCCA	DCI	PLE	PLE-LSTM	FUN(KFCV)	FUN(TAT)	UPPERBOUND
F_1^μ	RCV1/RCV2	.776 ± .052	.771 ± .050	.714 ± .061	.616 ± .065	.770 ± .052	.696 ± .060	.574 ± .113	.801† ± .044	**.802** ± .041	–
	JRC-Acquis	.559 ± .012	**.594** ± .016	.557 ± .024	.357 ± .023	.510 ± .014	.478 ± .061	.378 ± .041	.581 ± .010	.587 ± .009	.707
F_1^M	RCV1/RCV2	.467 ± .083	.490 ± .077	.471 ± .074	.385 ± .079	.485 ± .070	.453 ± .060	.302 ± .115	.512 ± .067	**.534** ± .066	–
	JRC-Acquis	.340 ± .017	**.411** ± .027	.379 ± .034	.206 ± .018	.317 ± .012	.300 ± .065	.182 ± .030	.356 ± .013	.399 ± .013	.599
K^μ	RCV1/RCV2	.690 ± .074	.696 ± .069	.659 ± .075	.550 ± .073	.696 ± .065	.644 ± .070	.515 ± .127	.731 ± .058	**.760** ± .052	–
	JRC-Acquis	.429 ± .015	.476 ± .020	.453 ± .029	.244 ± .022	.382 ± .016	.429 ± .050	.292 ± .046	.457 ± .012	**.490** ± .013	.632
K^M	RCV1/RCV2	.417 ± .090	.440 ± .086	.434 ± .080	.358 ± .088	.456 ± .082	.466 ± .073	.280 ± .118	.482 ± .075	**.506** ± .073	–
	JRC-Acquis	.288 ± .016	.348 ± .025	.330 ± .034	.176 ± .017	.274 ± .013	.349†† ± .047	.170 ± .032	.328 ± .013	**.365** ± .014	.547

Table 1. Multilabel PLC results; each cell indicates the value for the effectiveness measure and the standard deviation across the 10 runs. A greyed-out cell with a value in **boldface** indicates the best method (with the exclusion of UPPERBOUND). Superscripts † and †† denote the method (if any) whose score is not statistically significantly different from the best one at $\alpha = 0.05$ (†) or at $\alpha = 0.001$ (††).

the $\phi^2(d_l)$ representations for the training examples), the resulting computational cost would be severe.

Adhering to established practices in text classification we use two different kernels depending on the characteristics of the feature space. For all classifiers operating in a high-dimensional and sparse feature space (i.e., UPPERBOUND, LRI, the language-dependent classifiers of NAÏVE, plus the base classifiers of the two funnelling methods) we use the linear kernel, while we adopt the RBF kernel when the feature space is low-dimensional and dense (i.e., for CLESA, KCCA, DCI, PLE, and the meta-classifier of the two funnelling methods).

For the two funnelling methods we use the probability calibration algorithm implemented within `scikit-learn` and originally proposed by Platt [32], which consists of using, as the mapping function f, a logistic function

$$\Pr(c|d) = \frac{1}{1 + e^{\alpha h(d,c)+\beta}} \tag{10}$$

and choosing the parameters α and β in such a way as to minimize (via k-fold cross-validation) the negative log-likelihood of the training data.

5 RESULTS

5.1 Multilabel PLC experiments

Table 1 shows our multilabel PLC results. In this table (and in all the tables of the next sections) each reported value represents the average effectiveness across the 10 random versions of each dataset (see Sections 4.1.1 and 4.1.2) and (with the exception of the UPPERBOUND values, which are computed on English test data only) across the $|\mathcal{L}|$ languages in the dataset. We report results for eight combinations of (a) two datasets (RCV1/RCV2 and JRC-Acquis), (b) two evaluation measures (F_1 and K), and (c) two different ways of averaging the measure across the $|C|$ classes of the dataset (micro- and macro-averaging).

The results clearly indicate that our two funnelling methods perform very well. In particular, FUN(TAT) is the best performer in 6 out of 8 combinations of dataset, evaluation measure, averaging method, always outperforming all competitors in terms of the K measure and on the RCV1/RCV2 dataset. The only exception to this superiority is recorded for F_1^μ and F_1^M on the JRC-Acquis dataset, where LRI is the best method; note, however, that in these cases LRI outperforms FUN(TAT) only by a moderate margin, while in the previously discussed 6 cases the superiority of FUN(TAT) is more

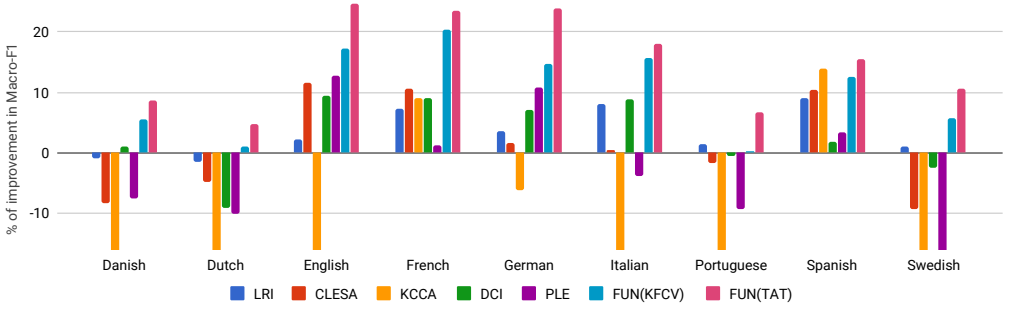

Fig. 2. Per-language percentage improvement in F_1^M with respect to each NAÏVE monolingual classifier in RCV1/RCV2. Some methods (notably: KCCA and PLE) sometimes exhibit deteriorations so large that they would be difficult to display in full; in these cases, bars are truncated at approximately -15% deterioration.

marked. In 8 out of 8 cases FUN(TAT) outperforms NAÏVE, CLESA, KCCA, DCI, PLE, and PLE-LSTM, almost always by a very wide margin.

The experiments also indicate that the simpler FUN(TAT) is consistently better than FUN(KFCV), with the former outperforming the latter in all 8 cases. Together with the fact that FUN(TAT) is markedly cheaper (by a factor of $(k+1)$) to train than FUN(KFCV), this makes FUN(TAT) our method of choice.

As already mentioned, the results displayed in Table 1 are averages across the $|\mathcal{L}|$ languages in the dataset. Analysing the results in a finer-grained way (that is, on a language-by-language basis) shows a further interesting fact: FUN(TAT) and FUN(KFCV) are the only systems that outperform the NAÏVE baseline *in every case*, i.e., for each language, dataset, evaluation measure, and averaging method (micro- or macro-). An example of this fact is shown in Figure 2, which displays the percentage improvement (in terms of F_1^M) obtained by the various methods with respect to the NAÏVE baseline for the various languages on the RCV1/RCV2 dataset. The figure shows that CLESA, DCI, KCCA, PLE, and even LRI (according to Table 1, the best competitor of funnelling methods), perform worse than NAÏVE for some languages, while both FUN(TAT) and FUN(KFCV) outperform NAÏVE for all languages. PLE-LSTM is not included in this plot since it always underperforms NAÏVE by such a large margin that including it in the plot would substantially hinder the visualization of the other results. FUN(TAT) thus proves not only the best method of the lot, but also the most stable.

That KCCA underperforms CLESA on most languages might be explained by the reduction in the number of Wikipedia articles that KCCA has observed (for the reasons discussed in Section 4.4) during training with respect to CLESA. Concerning PLE, instead, it is immediate to observe that it does not perform well, in many cases underperforming the NAÏVE baseline. A possible reason for this might reside in the fact that PLE was originally devised for (and showed good performance on) *sentence* classification; it is easy to conjecture that, when the units of classification are (as here) linguistic objects much longer than sentences, a method that just computes averages across word embeddings might introduce more noise than information. Regarding PLE-LSTM, we conjecture that its very bad performance might be explained by two facts. First, many words from different languages are not covered in the pre-trained multilingual embeddings; those words, that are instead

initialized with zero-embeddings[11], might affect negatively the entire optimization procedure. Second, it is very likely that the training set for each language is too small for a deep model to find meaningful cross-lingual patterns, thus making the classifier suffer from noisy information.

Incidentally, Figure 2 shows that the language on which Fun(tat) obtains the highest F_1^M improvement on RCV1/RCV2 with respect to the Naïve baseline, is English (in Table 4 we show this fact to hold in RCV1/RCV2 irrespectively of evaluation measure and averaging method). This shows that PLC techniques, and funnelling techniques in particular, can also benefit languages that are often considered "easy" (since they have historically received more attention than others from the research community), and for which obtaining improvements is thus considered harder.

An interesting observation we can make by observing Table 1 is that (a) UpperBound always works better than Fun(tat) and Fun(kfcv), and (b) Fun(tat) and Fun(kfcv) always work better than Naïve. Fact (a) indicates that the standard "bag of words", content-based representations which UpperBound uses work better than the representations based on posterior probabilities that Fun(tat) and Fun(kfcv) use, because UpperBound, Fun(tat) and Fun(kfcv) use exactly the same training examples (i.e., the examples in $\bigcup_{i=1}^{|\mathcal{L}|} Tr_i\}$), although represented differently. However, fact (b) shows that the inferior quality of the latter representations is more than compensated by the availability of many additional training examples, since Naïve uses a small subset ($|\mathcal{L}|$ times smaller) of the set of training examples that Fun(tat) and Fun(kfcv) use.

5.2 Multilabel monolingual and binary polylingual experiments

As discussed in Section 3.2, we conjecture that the good performance obtained by funnelling in the multilabel PLC experiments partly derives from the fact that the stochastic dependencies between the classes are brought to bear, and partly derives from the ability of funnelling to leverage training data written in language λ^s for classifying the data written in language λ^t. In order to verify if both factors indeed contribute to multilabel PLC, we run multilabel monolingual experiments and binary polylingual experiments.

In our multilabel monolingual experiments a funnelling system tackles a single language λ_i, i.e., there is just one 1st-tier multilabel classifier h_i^1 and the meta-classifier is trained only from the documents in Tr_i (instead of all the documents in $\bigcup_{i=1}^{|\mathcal{L}|} Tr_i$, as was the case in Section 5.1). (Note that, in this particular setting, stacking and funnelling coincide, as there is no heterogeneity in the data.) With such a setup, any improvement with respect to the Naïve baseline can only be due to the fact that funnelling brings to bear the stochastic dependencies between the classes. We run multilabel monolingual experiments independently for all the $|\mathcal{L}|$ languages in the dataset. The results (reported as averages across these $|\mathcal{L}|$ languages) are displayed in Column B of Table 2.

In our binary polylingual experiments, instead, a funnelling system tackles a single class, i.e., the $\phi^2(d_u)$ vectors fed to the meta-classifier only consist of one posterior probability (instead of $|C|$ posterior probabilities, as was the case in Section 5.1), so that any improvement with respect to the Naïve baseline can only be due to the ability of funnelling to leverage training data written in language λ^s for classifying the data written in language λ^t. We run binary polylingual experiments independently for all the $|C|$ classes in the dataset. The results are displayed in Column C of Table 2.

Note that in these experiments (i) we do not run LRI, CLESA, DCI, and PLE, since our only goal here is to assess where the improvements of funnelling with respect to the Naïve baseline come from; (ii) we only run Fun(tat) since its superiority with respect to Fun(kfcv) has already been ascertained in a fairly conclusive way in Section 5.1; (iii) in Table 2 (as, for that matter, in all other

[11] We have tested other approaches including random initialization, or replacing them with a language-specific *unknown* token. None of them effectively help to improve the results.

		A	B	C	D
		NAÏVE Binary MonoLin	FUN(TAT) MultiLab MonoLin	FUN(TAT) Binary PolyLin	FUN(TAT) MultiLab PolyLin
F_1^μ	RCV1/RCV2	.776 ± .052	.800†† ± .002	.801†† ± .002	**.802** ± .041
	JRC-Acquis	.559 ± .012	.577 ± .002	**.589** ± .002	.587†† ± .009
F_1^M	RCV1/RCV2	.467 ± .083	.526 ± .013	.532† ± .014	**.534** ± .066
	JRC-Acquis	.340 ± .017	.369 ± .002	.395†† ± .003	**.399** ± .013
K^μ	RCV1/RCV2	.690 ± .074	.747 ± .003	.757 ± .004	**.760** ± .052
	JRC-Acquis	.429 ± .015	.454 ± .002	.487†† ± .002	**.490** ± .013
K^M	RCV1/RCV2	.417 ± .090	.492 ± .013	.505† ± .014	**.506** ± .073
	JRC-Acquis	.288 ± .016	.325 ± .003	.359 ± .003	**.365** ± .014

Table 2. FUN(TAT) results for multilabel monolingual classification (Column B) and binary polylingual classification (Column C). The results in Columns A and D are from Table 1, and are reported here only for ease of comparison. The notational conventions are the same as in Table 1.

tables in this paper) the results reported in the 4 columns for the same row are all comparable with each other, since the training set and the test set are the same in all 4 cases.

The results of Table 2 suggest the following observations:

(1) Using FUN(TAT) in order to bring to bear the stochastic dependencies between different classes is useful, as witnessed by the fact that the figures for the multilabel monolingual setup are always higher than the corresponding figures for the NAÏVE baseline.
(2) Using FUN(TAT) in order to leverage training data written in one language for classifying the data written in other languages, is also useful, as witnessed by the fact that the figures for the binary polylingual setup are always higher than the corresponding figures for the NAÏVE baseline.
(3) The two observations above are confirmed by the fact that the figures for the multilabel polylingual setup are (almost always) higher than the figures for both the multilabel monolingual and the binary polylingual setups. In other words, *both* factors contribute to the fact that FUN(TAT) in the multilabel polylingual setup improves on the NAÏVE baseline.
(4) While both factors do contribute, it is also clear that the bigger contribution comes not from the stochastic dependencies between different classes, but from the training data in other languages, as witnessed by the fact that the figures for the multilabel polylingual setup are much closer to the binary polylingual ones than to the multilabel monolingual ones.

5.3 Learning curves for the under-resourced languages

As we have mentioned in the introduction, PLC techniques are especially useful when we need to perform text classification for under-resourced languages, i.e., languages for which only a small number of training documents are available. In this section we provide the results of experiments aimed at showing how funnelling performs in such situations. We simulate these scenarios by testing, on the λ_i test data, a FUN(TAT) system trained on all the training data for the languages in $\mathcal{L}/\{\lambda_i\}$ and on variable fractions of the training data for λ_i, which thus plays (especially when these fractions are small) the role of the under-resourced language. When this fraction is 0% of the lot, this corresponds to the cross-lingual setting; when it is 100% of the lot, this corresponds to the

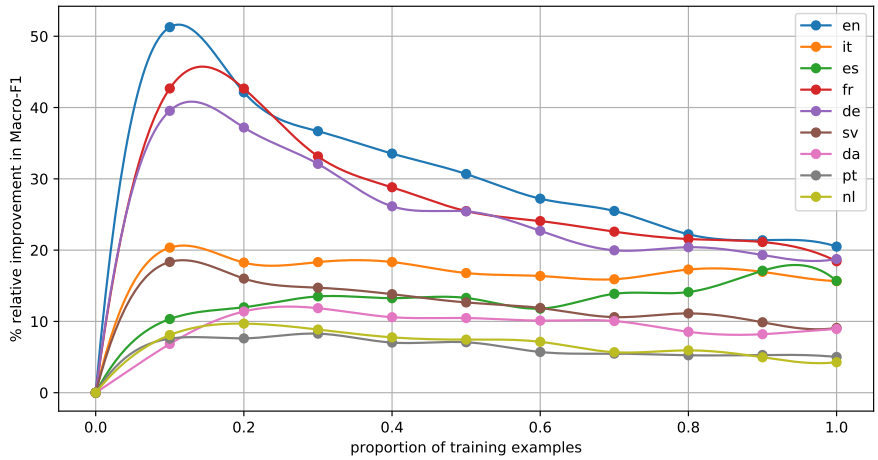

Fig. 3. Relative improvement in terms of F_1^M on the RCV1/RCV2 dataset obtained by using FUN(TAT) with respect to using NAÏVE. Values on the x axis are the fractions of Tr_i that are used for training.

setup we have studied in Section 5.1. In our experiments we generate these fractions by randomly removing increasing amounts of data from the training set, so that the training sets for the smaller fractions are proper subsets of those for the larger fractions. Like for all other experiments in this paper, the results we report are averages across the 10 random trials discussed at the end of Sections 4.1.1 and 4.1.2.

Figure 3 shows, for the RCV1/RCV2 dataset and the F_1^M measure (the cases of JRC-Acquis and/or the other measures show similar patterns), the improvements which are obtained on the test sets of the individual languages λ_i as a function of the fraction of the training data Tr_i used. There are three main observations that we can make: (i) for each language λ_i and each fraction of training data used, the variation in accuracy is always positive, i.e., there is always an improvement in accuracy (and never a deterioration) as a result of using funnelling; (ii) some languages benefit more than others (in our case, English, French, and German stand out in this respect); (iii) the improvements are more marked when small fractions of λ_i training data are used. Fact (ii) will be the subject of detailed study in Section 5.4. As for Fact (iii), this is intuitive after all, since it is when the accuracy of a monolingual classifier is low (as it presumably is when it has been trained from few labelled data) that the margins of improvement resulting from the contributions of other languages are high.

5.4 Which languages contribute/benefit most?

In this section we present "ablation" experiments in which we attempt to understand (a) which languages contribute most, and (b) which languages benefit most, in terms of the classification effectiveness that can be obtained via FUN(TAT) in multilabel PLC. In order to do this, for each pair of languages $\lambda^s, \lambda^t \in \mathcal{L}$ we classify the λ^t test data via (1) a FUN(TAT) system trained on $\mathcal{L}/\{\lambda^s\}$ training data, and (2) a FUN(TAT) system trained on $\mathcal{L}$ training data. The improvement $i(\lambda^s, \lambda^t)$ observed in switching from (1) to (2) is a measure of the contribution that λ^s training data offer to classifying λ^t data, or (said another way) of the benefit that the classification of λ^t data obtains from the presence of λ^s training data. Similarly to what we have done in Section 5.3, in

		EN	IT	ES	FR	DE	SV	DA	PT	NL	FI	HU
F_1^μ	RCV1/RCV2	+0.08%	+0.68%	+0.34%	+0.49%	+0.03%	**+2.25%**	+0.06%	+0.41%	+0.18%	–	–
	JRC-Acquis	-0.11%	+2.85%	-0.20%	+0.67%	+0.01%	-0.56%	-0.12%	+2.67%	**+3.35%**	+0.03%	+1.84%
F_1^M	RCV1/RCV2	-0.05%	+0.36%	-0.00%	+0.11%	+0.04%	+0.75%	+0.17%	**+1.19%**	+0.82%	–	–
	JRC-Acquis	-0.64%	+5.98%	-0.95%	+0.83%	-0.45%	-10.23%	-0.37%	+3.61%	**+6.23%**	-0.60%	+3.76%
K^μ	RCV1/RCV2	+0.70%	+1.52%	+0.99%	+0.41%	+1.12%	**+7.71%**	+0.74%	+2.91%	+1.65%	–	–
	JRC-Acquis	+0.80%	+7.85%	+1.07%	+3.63%	+0.63%	+2.67%	+0.16%	+7.78%	**+8.83%**	+1.85%	+5.90%
K^M	RCV1/RCV2	+0.39%	+1.03%	+0.45%	+0.39%	+0.60%	**+3.55%**	+0.46%	+2.81%	+2.07%	–	–
	JRC-Acquis	+0.99%	+10.97%	+1.37%	+4.74%	+0.66%	-1.23%	+0.15%	+9.20%	**+11.65%**	+2.30%	+8.36%

Table 3. Average contribution (across languages $\lambda^t \in \mathcal{L}/\{\lambda^s\}$) provided by λ^s training data to classifying λ^t test data via FUN(TAT). A greyed-out cell with a value in **boldface** indicates the language that has contributed most.

all these experiments we adopt an "under-resourced language" setting and use only 10% of the λ^t training examples. Note that the notion of "improvement in effectiveness" mentioned above depends on which measure of effectiveness (among the four we have employed in this paper) we use as reference.

Displaying all the $|\mathcal{L}| \times |\mathcal{L}|$ individual $i(\lambda^s, \lambda^t)$ results would probably not allow significant insights to be obtained. However, in our multilabel PLC context they can be aggregated so as to measure

(1) which languages contribute most to the classification of data in other languages; we compute the contribution $\alpha(\lambda^s)$ of language λ^s as the average value of $i(\lambda^s, \lambda^t)$ across all $\lambda^t \in \mathcal{L}/\{\lambda^s\}$;
(2) which languages benefit most from the presence of training data in other languages; we compute the benefit $\beta(\lambda^t)$ that language λ^t obtains as the average value of $i(\lambda^s, \lambda^t)$ across all $\lambda^s \in \mathcal{L}/\{\lambda^t\}$.

These results are reported in Tables 3 and 4. Rather than commenting on the individual cases, one interesting question we may ask ourselves is: what are the factors that make a language contribute more, or benefit more, within a funnelling system for PLC? Are there interesting correlations between these contributions / benefits and other measurable characteristics of the individual languages? Note that all languages have the same number of training examples (and they also have the same number of test examples), both in RCV1/RCV2 and JRC-ACQUIS, so (even considering what we say in Footnote 6) language frequency is unlikely to be a factor in our experiments.

A first conjecture we have tested is if the contribution $\alpha(\lambda^s)$ is positively correlated with the accuracy of the NAÏVE classifier for language λ^s as computed on λ^s test data (we here denote this accuracy as $F_1^M(\text{NAÏVE}(\lambda^s))$).[12] This conjecture would seem sensible, since we would expect the contribution of a language to be high when its language-specific training data are high-quality (which is witnessed by the fact that a classifier trained on them is capable of delivering high accuracy). We measure correlation via the *Pearson Correlation Coefficient* (PCC), noted as $\rho(X, Y)$; its values range on [-1,+1], with -1 indicating perfect negative correlation, +1 indicating perfect positive correlation, and 0 indicating total lack of correlation. The above conjecture proves essentially correct, since the resulting value of PCC is $\rho(\alpha(\lambda^s), F_1^M(\text{NAÏVE}(\lambda^s))) = 0.788$ (with a p-value of 0.011), which indicates high correlation.[13]

[12] As in Section 5.3, as the measure of accuracy we here employ F_1^M in computing both $\alpha(\lambda^s)$ and the accuracy of the NAÏVE classifier for language λ^s; the other measures used in this paper display similar results.
[13] For PCC, the p-value indicates the probability that two random variables that have no correlation generate a sample characterized by a value of PCC at least as extreme as the one of the present sample.

		EN	IT	ES	FR	DE	SV	DA	PT	NL	FI	HU
F_1^μ	RCV1/RCV2	**+1.70%**	+0.76%	+0.39%	+0.95%	+0.72%	+0.08%	-0.26%	+0.20%	-0.01%	–	–
	JRC-Acquis	+0.17%	+1.22%	+1.59%	+1.31%	+2.20%	-0.45%	+1.16%	+0.53%	+0.61%	**+2.27%**	-0.17%
F_1^M	RCV1/RCV2	**+2.98%**	+0.27%	+0.13%	-0.12%	+0.56%	-0.06%	-0.62%	+0.21%	+0.04%	–	–
	JRC-Acquis	+1.13%	**+2.12%**	+1.08%	+1.22%	+1.55%	-2.16%	+1.05%	+0.57%	+1.17%	+0.75%	-1.31%
K^μ	RCV1/RCV2	**+3.33%**	+2.73%	+1.81%	+2.40%	+2.41%	+1.26%	+0.12%	+2.60%	+1.10%	–	–
	JRC-Acquis	+3.06%	+4.45%	+4.21%	+4.47%	+4.85%	+1.89%	+3.34%	+3.01%	+3.14%	**+5.62%**	+3.15%
K^M	RCV1/RCV2	**+4.68%**	+1.37%	+0.77%	+1.01%	+2.60%	+0.51%	-0.25%	+0.80%	+0.25%	–	–
	JRC-Acquis	+4.85%	**+5.89%**	+4.29%	+5.28%	+5.15%	+1.57%	+3.57%	+4.32%	+4.69%	+5.45%	+4.11%

Table 4. Average benefit (across languages $\lambda^s \in \mathcal{L}/\{\lambda^t\}$) obtained from the presence of λ^s training data in classifying λ^t test data via FUN(TAT). A greyed-out cell with a value in **boldface** indicates the language that has benefited most.

A second conjecture we have tested is if the benefit $\beta(\lambda^t)$ is negatively correlated with the accuracy of the NAÏVE classifier for language λ^t (once trained with only 10% of the λ^t training examples, which is the setting we have adopted in this section) as tested on λ^t test data. This conjecture would also seem sensible, since we might expect the benefit $\beta(\lambda^t)$ to be higher when the effectiveness of NAÏVE on language λ^t is lower, since in this case the margins of improvement are higher. In this case too, the conjecture proves essentially correct, since the resulting value of PCC is $\rho(\beta(\lambda^t), F_1^M(\text{NAÏVE}(\lambda^t))) = -0.605$ (p-val 0.08411), which indicates substantial negative correlation.

5.5 Can we do without calibration?

As remarked in Section 3, one of the aspects that contributes more substantially to the computational cost of funnelling systems is probability calibration. The reason is that, as also remarked in Section 4.5, calibration consists in finding the optimal parameters of Equation 10 through an extensive search within the space of parameter values. It is thus of some interest to study whether we can do without calibration at all, and what the effect of this would be. We have thus run FUN(TAT) experiments in order to compare three alternative courses of action:

(1) NOPROB: Renounce to converting classification scores into posterior probabilities. In this setting, a FUN(TAT) system is set up in which the metaclassifier (i) is trained with training documents represented by vectors $S(d_l)$ of classification scores, and, (ii) once trained, classifies documents represented by vectors $S(d_u)$ of classification scores.
(2) NOCALIB: Convert classification scores into posterior probabilities, but renounce to calibrate them. This corresponds to employing a version of FUN(TAT) where, in place of the logistic function of Equation 10, we use a non-parametric version of it, which corresponds to Equation 10 with parameters α and β fixed to 1 and 0, respectively.
(3) CALIB: Employ the usual version of FUN(TAT) as defined in Section 3.1.

In Table 5 we report the results of running these three alternative systems; the experimental setting is the same of Section 5.1, and the results of Columns "NAÏVE" and "CALIB" of Table 5 indeed coincide with those of Columns "NAÏVE" and "FUN(TAT)" of Table 1.

One fact that emerges from these results is that the standard CALIB setting always delivers the best performance, which is unsurprising. A second fact that emerges is that the NOCALIB setting is always inferior to the NOPROB setting. This is surprising, since we might have conjectured NOCALIB to outperform NOPROB, due to the fact that NOCALIB makes the outputs of the different base classifiers more comparable among each other (by mapping them all into the [0,1] interval) than the outputs used by NOPROB; this finding *de facto* rules out NOCALIB from further consideration.

Something that is much less clear, instead, is how NOPROB performs relative to NAÏVE and to the standard CALIB setting. In some cases NOPROB performs very well, almost indistinguishably from

		NAÏVE	NOPROB	NOCALIB	CALIB
F_1^μ	RCV1/RCV2	.776 ± .052	.796 ± .045	.789 ± .048	**.802 ± .041**
	JRC-Acquis	.559 ± .012	.585††† ± .012	.578 ± .012	**.587 ± .009**
F_1^M	RCV1/RCV2	.467 ± .083	.463 ± .082	.443 ± .086	**.534 ± .066**
	JRC-Acquis	.340 ± .017	.376 ± .021	.366 ± .015	**.399 ± .013**
K^μ	RCV1/RCV2	.690 ± .074	.737 ± .062	.716 ± .069	**.760 ± .052**
	JRC-Acquis	.429 ± .015	.478† ± .018	.465 ± .015	**.490 ± .013**
K^M	RCV1/RCV2	.417 ± .090	.428 ± .087	.406 ± .091	**.506 ± .073**
	JRC-Acquis	.288 ± .016	.338† ± .022	.325 ± .016	**.365 ± .014**

Table 5. Multilabel PLC results with alternative FUN(TAT) settings. Notational conventions are as in Table 1.

	NAÏVE	LRI	CLESA	KCCA	DCI	PLE	PLE-LSTM	FUN(KFCV)	FUN(TAT)
RCV1/RCV2	537 ± 69	5,506 ± 603	28,508 ±5351	18,204 ± 15	344 ± 51	1,293 ± 6	559 ± 103	1,041 ± 112	**215** ± 16
	6 ± 0.3	91 ± 3	575 ± 10	264 ± 7	9 ± 0.2	55 ± 1	**3** ± 0.1	13 ± 0.5	11 ± 0.4
JRC-Acquis	6,005 ± 1,351	67,571 ± 2,070	63,497 ± 2,880	57,563 ± 241	4,888 ± 1,136	**4,435** ± 25	26,991 ± 915	13,127 ± 2,428	4,987 ± 208
	84 ± 2	1,713 ± 6	4,049 ± 123	1,372 ± 67	253 ± 3	874 ± 11	**6** ± 0.4	312 ± 4	278 ± 2

Table 6. Computation times (in seconds); 1st rows indicate training times while 2nd rows report testing times.

CALIB (see F_1^μ results for JRC-Acquis), but in other cases it even performs worse than the NAÏVE baseline, and dramatically worse than CALIB (see F_1^M results for RCV1/RCV2).

All in all, these results confirm the theoretical intuition that performing a full-blown probability calibration is by far the safest option, and the one guaranteed to deliver the best results in all situations.

5.6 Efficiency

Table 6 reports training times and testing times for all the methods discussed in this paper, as clocked on our two datasets; each reported value is the average value across the 10 random trials. The experiments were run on a machine equipped with a 12-core processor Intel Core i7-4930K at 3.40GHz with 32 GB of RAM under Ubuntu 16.04 (LTS). For PLE-LSTM, the times reported correspond to our Keras implementation running on a Nvidia GeForce GTX 1080 equipped with 8 GB of RAM. We limit our analysis to the multilabel PLC setup of Section 5.1 (thus skipping the discussion of the setups of Sections 5.2 and 5.3) (i) since multilabel PLC is the most interesting context, and (ii) since for the setups discussed in Sections 5.2 and 5.3 we have run only FUN(TAT) and NAÏVE.

The most interesting fact that emerges from Table 6 is that the superior accuracy of FUN(TAT) does not come at a price. Indeed, FUN(TAT) often turns out to be one of the most efficient, or sometimes *the* most efficient, among the methods we test; in particular, both at training time and testing time it is one order of magnitude faster than LRI, its most important competitor. FUN(KFCV) is, as previously observed, much more expensive to train than FUN(TAT), due to the much higher number of training and probability calibration rounds that it requires. CLESA is clearly the most inefficient of all methods, which is explained by the fact that each (labelled or unlabelled) document

requires one document similarity computation for each feature in its vectorial representation. The higher training-time efficiency of FUN(TAT) with respect to NAÏVE is certainly also due to the fact that, as mentioned in Section 4.5, we do not perform any optimization of the C parameter for the base classifiers of FUN(TAT), while we do for the classifiers of NAÏVE; should we perform this parameter optimization the computational cost of FUN(TAT) would certainly increase, but so probably would also the differential in effectiveness between FUN(TAT) and all the other baselines.

Note that the most efficient method in testing mode is PLE-LSTM, especially in the case of JRC-Acquis, where it is one order of magnitude faster than the 2nd fastest method (NAÏVE). The reasons are twofold: (a) as noted above, the PLE-LSTM experiments have been run on hardware different from the hardware used for all the other experiments, so comparisons are difficult to make; (b) in models trained via deep learning, such as PLE-LSTM, testing reduces to a simple forward pass through the network connections, something which can be performed very quickly by exploiting the massive parallelism offered by modern GPUs.

6 CAN FUNNELLING BE USED IN THE CROSS-LINGUAL SETTING?

The experiments we have discussed so far have assumed a *polylingual* text classification setting, i.e., one in which there is a non-null number of training examples for each of the target languages, and in which the training examples for the source languages have thus the goal of *improving* the accuracy of the classifiers generated from the training examples of the target languages. We might wonder whether funnelling can also be used in a *cross-lingual* setting, i.e., one in which there are no training examples for the target languages, and in which the training examples for the source languages would have the goal of allowing to generate classifiers for the target languages that could otherwise not be generated at all.

Unfortunately, the answer is no. To see why, for simplicity let us discuss FUN(TAT) (the case of FUN(KFCV) is analogous). If there are no positive training documents for pair (λ_i, c_j), this means that (as noted in Section 4.5) the base classifier h_i^1 generated from the negative examples only (i.e., from the examples in λ_i that are positive for some other class in $C/\{c_j\}$) is a trivial rejector for c_j, i.e., one that only returns scores $h_i^1(d_u, c_j) = 0$ for all unlabeled documents d_u written in language λ_i. By definition, the calibration function turns all these scores into posterior probabilities $\Pr(c_j|d_u) = 0$. As a result, when the negative training examples are reclassified by h_i^1 for generating vectorial representations that contribute to training the metaclassifier, these negative training examples originate vectors that contain a 0 for class c_j. Since these are all negative examples, the metaclassifier is trained to interpret a value of 0 in the vector position corresponding to c_j as a perfect predictor that the document does not belong to c_j. As a result, when an unlabelled document in language λ_i is classified, the base classifier returns a value $h_i^1(d_u, c_j) = 0$, which is converted into a posterior probability $\Pr(c_j|d_u) = 0$, which is thus interpreted as unequivocally indicating that d_u does not belong to c_j, independently of the contributions coming from classes other than c_j and languages other than λ_i. The entire 2-tier classifier is then a trivial rejector for pair (λ_i, c_j).[14] This shows that funnelling is unsuitable for dealing with the scenario in which there are no training examples for the target languages.

This problem has prompted us to devise ways of enabling funnelling to also operate in "zero-shot mode" (i.e., on documents expressed in languages for which no training documents are available). The basic idea is to add a "zero-shot classifier" $h^1_{(|\mathcal{L}|+1)}$ (which for notational simplicity we denote

[14]Note that this is confirmed by the experiments plotted in Figure 3, where for $x = 0$ it holds that $F_1^M = 0$ for all languages λ_i. In fact, when there are no training examples for the target language ($x = 0$) the entire 2-tier classifier is, as observed above, a trivial rejector, which means that TP is 0 and, as a consequence, F_1 is 0 too, as clearly visible for all plots in the figure.

by h^1_z) to the 1st-tier classifiers, i.e., a classifier that is to be invoked whenever a document written in any language different from the ones in $\mathcal{L}$ (i.e., from the languages for which training examples do exist) needs to be classified. This means that the 2nd-tier classifier is trained also on (and also receives as input) the posterior probabilities returned by h^1_z, which thus needs to be a well calibrated classifier. Note that this modification fits smoothly into the framework, since funnelling makes very few assumptions about the characteristics of the base classifiers. For simplicity, we here derive the adaptation for Fun(tat); the case of Fun(kfcv) is similar.

More formally, let $\mathcal{L}$ be a set of languages for which labelled training examples are available. In this new variant of the funnelling system, in the 1st tier there are (as usual) $|\mathcal{L}|$ language-specific classifiers $h^1_1, \ldots, h^1_{|\mathcal{L}|}$, plus one classifier h^1_z trained (according to some method yet to be specified) on *all* the training examples in any of the languages in $\mathcal{L}$. For each training document d_l in language λ_i, *two* vectorial representations are generated that are used in training the 2nd-tier classifier h^2, i.e., the vector of posterior probabilities

$$(f_i(h^1_i(d_l, c_1)), \ldots, f_i(h^1_i(d_l, c_{|C|})))$$

from the language-dependent classifier h^1_i, and the vector of posterior probabilities

$$(f_z(h^1_z(d_l, c_1)), \ldots, f_z(h^1_z(d_l, c_{|C|})))$$

from the zero-shot classifier h^1_z. Therefore, h^2 is trained on twice the number of $|C|$-dimensional vectors with respect to the one we considered in the previous sections.

When a new unlabelled document d_u expressed in language λ is submitted for classification, two scenarios are possible:

(1) $\lambda \in \mathcal{L}$: this case reduces to funnelling as discussed in the previous sections, that is, (i) the document is first represented in its corresponding language-specific feature space, (ii) a vector of posterior probabilities is then obtained using the corresponding language-specific 1st-tier classifier, and (iii) the 2nd-tier classifier h^2 takes the final decision;
(2) $\lambda \notin \mathcal{L}$: in this case, (i) the document is first represented in the feature space of h^1_z, (ii) a vector of posterior probabilities is then obtained using the calibrated 1st-tier classifier h^1_z, and (iii) the 2nd-tier classifier h^2 takes the final decision.

CLESA, PLE, and PLE-LSTM are possible methods by means of which the representations $\phi^1_z(d)$ in the feature space of h^1_z can be obtained. For example, PLE trains a classifier on representations of the documents consisting of averages of polylingual word embeddings. Since polylingual word embeddings are aligned across languages [7], the same classifier would, in principle, be capable of classifying a document written in any language λ (possibly with $\lambda \notin \mathcal{L}$) for which pre-trained and aligned word embeddings are available. Similar considerations enable CLESA to work with documents in languages not in $\mathcal{L}$, as long as a set of comparable Wikipedia articles are available for their language.

For our experiments we choose PLE as the method to generate the 1st-tier zero-shot classifier, because of the good trade-off between effectiveness and efficiency it has shown in our previous experiments. We call the resulting cross-lingual classification method Fun(tat)-PLE.

In order to test Fun(tat)-PLE we run experiments in which, experiment by experiment, we augment the set of languages for which training examples are available. In each new experiment, the training set of a new language is added, while the languages for which training data have not been added yet are dealt with by the zero-shot classifier. For example, after the third experiment, the training data for the three languages {da,de,en} have been added to the training set (we add languages following the alphabetical order). The test set is instead fixed, and always contains all test examples of all languages.

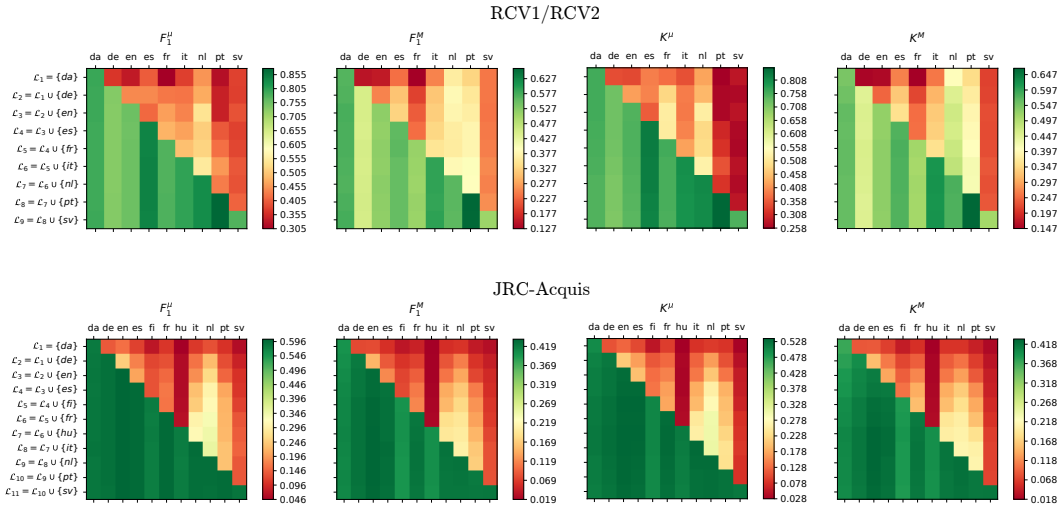

Fig. 4. Cross-lingual classification experiments using Fun(tat)-PLE in RCV1/RCV2 (top) and JRC-Acquis (bottom) for the four evaluation measures (from left to right) F_1^{μ}, F_1^{M}, K^{μ}, and K^{M}. In each square matrix, columns represent test languages, while rows represent training sets with an increasing (from top to bottom) number of languages.

The results of our experiments are displayed in graphical form in Figure 4, where colours are used instead of numerical data in order to make patterns and trends more evident. Each of the 8 square matrixes of coloured cells represents the experiments performed on one of our 2 datasets and using one of our 4 evaluation measures; each cell in a matrix represents the accuracy obtained using the training data for a given group of languages (indicated on the row) and the test data for a given language (indicated on the column). In each such matrix, the lower triangular matrix reflects the classification outcomes on test languages which are represented in the training set; because of this, accuracy results are typically high (green). The upper triangular matrix represents the outcomes for languages that are *not* represented in the training data, which thus tend to obtain lower scores (red).

One clear pattern that emerges from Figure 4 is that the piecemeal addition of languages to the training set improves the classification accuracy for the yet unseen (i.e., not represented in the training set) languages, as witnessed by the gradual change in colour through columns, from dark red on top to lighter red towards the bottom.

Notwithstanding this, a similar improvement does not clearly emerge for the already seen languages, i.e., the addition of languages to the training set does not seem to boost the classification accuracy for the languages already represented in the training set. However, such an improvement does exist in the "pure" version of Fun(tat), as verified and discussed in Sections 5.2, 5.3, and 5.4. A possible explanation for this anomaly might be a negative side-effect introduced by the h_z^1 classifier into the meta-classifier.

In sum, the experiments discussed in this section seem to indicate (i) that funnelling, as a framework, can indeed be adapted to cross-lingual classification, and (ii) that better ways of combining the posterior probabilities returned by the 1st-tier classifiers should be investigated for the cross-lingual case. This is something we plan to do in future research.

7 CONCLUSION

This paper presents (i) a novel 2-tiered ensemble learning method for heterogeneous data, and (ii) the first (to the best of our knowledge) application of an ensemble learning method to multilingual (and more specifically: polylingual multilabel) text classification. While similar to stacked generalization, this ensemble learning method (that we dub "funnelling") is different from it because the base classifiers are specialized, each catering for a different type of objects characterized by its own feature space. In polylingual classification, this means that different base classifiers deal with documents written in different languages; funnelling makes it possible to bring them all together, so that the training examples for all languages in $\mathcal{L}$ contribute to the classification of all unlabelled documents, irrespectively of the language $\lambda \in \mathcal{L}$ they are written in.

One advantage of funnelling is that it is learner-independent; while in this paper we test it with SVMs as the learning method, it can be set up to use (i) any learning device that outputs non-binary classification scores (for the base classifiers), and (ii) any learning device that accepts numeric feature values as input (for the metaclassifier). An additional advantage of funnelling is that, unlike several other multilingual methods, it does not require external resources, either in the form of multilingual dictionaries, or machine translation services, or external parallel corpora.

The extensive experiments we have run on a comparable 9-language corpus (RCV1/RCV2) and on a parallel 11-language corpus (JRC-Acquis) against a number of state-of-the-art baseline methods, show that Fun(tat) (the better of two funnelling methods we have tested) (i) almost always outperforms all baselines, irrespectively of evaluation measure, averaging method, and dataset; (ii) delivers improvements over the naïve monolingual baseline more consistently (i.e., for all tested languages, datasets, evaluation measures, averaging methods) than any other baseline considered; and (iii) is among the most efficient tested methods, at both training time and testing time. All this has been confirmed across a range of experimental settings, i.e., binary or multilabel, monolingual or polylingual. The two main factors behind the success of funnelling in polylingual multilabel classification are (i) its ability to leverage the training examples written in any language in order to classify unlabelled examples written in any language, and (ii) its ability to leverage the stochastic dependencies between different classes.

Funnelling is useful whenever (i) the data to be classified comes in different types that require different feature representations, *and* (ii) despite these differences in nature, all data need to be classified under a common classification scheme C. We are currently testing funnelling in other such contexts, as in e.g., classifying images of products and textual descriptions of products under the same set C of product classes.

REFERENCES

[1] Georgios Balikas and Massih-Reza Amini. 2016. Multi-label, multi-class classification using polylingual embeddings. In *Proceedings of the 38th European Conference on Information Retrieval (ECIR 2016)*. Padova, IT, 723–728. https://doi.org/10.1007/978-3-319-30671-1_59

[2] Nuria Bel, Cornelis H. Koster, and Marta Villegas. 2003. Cross-lingual text categorization. In *Proceedings of the 7th European Conference on Research and Advanced Technology for Digital Libraries (ECDL 2003)*. Trondheim, NO, 126–139. https://doi.org/10.1007/978-3-540-45175-4_13

[3] Natalia Y. Bilenko and Jack L. Gallant. 2016. Pyrcca: Regularized kernel canonical correlation analysis in Python and its applications to neuroimaging. *Frontiers in Neuroinformatics* 10 (2016), 49. https://doi.org/10.3389/fninf.2016.00049

[4] Christopher M. Bishop. 2006. *Pattern Recognition and Machine Learning*. Springer, Heidelberg, DE.

[5] Leo Breiman. 1996. Bagging predictors. *Machine Learning* 24, 2 (1996), 123–140. https://doi.org/10.1007/bf00058655

[6] Philip K. Chan and Salvatore J. Stolfo. 1997. On the Accuracy of Meta-Learning for Scalable Data Mining. *Journal of Intelligent Information Systems* 8, 1 (1997), 5–28. https://doi.org/10.1023/A:1008640732416

[7] Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2018. Word Translation without Parallel Data. In *Proceedings of the 6th International Conference on Learning Representations (ICLR 2018)*.

Vancouver, CA.
[8] Oscar Day and Taghi M. Khoshgoftaar. 2017. A survey on heterogeneous transfer learning. *Journal of Big Data* 4 (2017), Article 17 (1–42). https://doi.org/10.1186/s40537-017-0089-0
[9] Morris H. DeGroot and Stephen E. Fienberg. 1983. The comparison and evaluation of forecasters. *The Statistician* 32, 1/2 (1983), 12–22. https://doi.org/10.2307/2987588
[10] Susan T. Dumais, Todd A. Letsche, Michael L. Littman, and Thomas K. Landauer. 1997. Automatic cross-language retrieval using latent semantic indexing. In *Working Notes of the AAAI Spring Symposium on Cross-language Text and Speech Retrieval*. Stanford, US, 18–24. https://doi.org/10.1007/978-1-4615-5661-9_5
[11] Saso Džeroski and Bernard Ženko. 2004. Is Combining Classifiers with Stacking Better than Selecting the Best One? *Machine Learning* 54, 3 (2004), 255–273. https://doi.org/10.1023/b:mach.0000015881.36452.6e
[12] Manaal Faruqui and Chris Dyer. 2014. Improving Vector Space Word Representations Using Multilingual Correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*. Gothenburg, SE, 462–471. https://doi.org/10.3115/v1/e14-1049
[13] Marc Franco-Salvador, Paolo Rosso, and Roberto Navigli. 2014. A Knowledge-based Representation for Cross-Language Document Retrieval and Categorization. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2014)*. Gothenburg, SE, 414–423. https://doi.org/10.3115/v1/e14-1044
[14] Yoav Freund and Robert E. Schapire. 1996. Experiments with a New Boosting Algorithm. In *Proceedings of the 13th International Conference on Machine Learning (ICML 1996)*. Bari, IT, 148–156.
[15] Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing Semantic Relatedness Using Wikipedia-based Explicit Semantic Analysis. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence (IJCAI 2007)*. San Francisco, US, 1606–1611.
[16] Juan José García Adeva, Rafael A. Calvo, and Diego López de Ipiña. 2005. Multilingual approaches to text categorisation. *European Journal for the Informatics Professional* 5, 3 (2005), 43–51.
[17] Shantanu Godbole and Sunita Sarawagi. 2004. Discriminative Methods for Multi-labeled Classification. In *Proceedings of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD 2004)*. Sydney, AU, 22–30. https://doi.org/10.1007/978-3-540-24775-3_5
[18] Stephan Gouws, Yoshua Bengio, and Greg Corrado. 2015. Bilbowa: Fast bilingual distributed representations without word alignments. In *Proceedings of the 32nd International Conference on Machine Learning (ICML 2015)*. Lille, FR, 748–756.
[19] Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850* (2013).
[20] David R. Hardoon, Sandor Szedmak, and John Shawe-Taylor. 2004. Canonical correlation analysis: An overview with application to learning methods. *Neural Computation* 16, 12 (2004), 2639–2664. https://doi.org/10.1162/0899766042321814
[21] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[22] Harold Hotelling. 1936. Relations between two sets of variates. *Biometrika* 28, 3/4 (1936), 321–377. https://doi.org/10.2307/2333955
[23] Alexandre Klementiev, Ivan Titov, and Binod Bhattarai. 2012. Inducing Crosslingual Distributed Representations of Words. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING 2012)*. Mumbai, IN, 1459–1474.
[24] Ludmila I. Kuncheva. 2004. *Combining Pattern Classifiers: Methods and Algorithms*. John Wiley & Sons, Hoboken, US.
[25] David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research* 5 (2004), 361–397.
[26] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Proceedings of the 27th Annual Conference on Neural Information Processing Systems (NIPS 2013)*. Lake Tahoe, US, 3111–3119.
[27] David Mimno, Hanna M. Wallach, Jason Naradowsky, David A. Smith, and Andrew McCallum. 2009. Polylingual topic models. In *Proceedings of the 7th Conference on Empirical Methods in Natural Language Processing (EMNLP 2009)*. Singapore, SN, 880–889. https://doi.org/10.3115/1699571.1699627
[28] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Distributional Correspondence Indexing for Cross-Lingual and Cross-Domain Sentiment Classification. *Journal of Artificial Intelligence Research* 55 (2016), 131–163.
[29] Alejandro Moreo, Andrea Esuli, and Fabrizio Sebastiani. 2016. Lightweight Random Indexing for Polylingual Text Classification. *Journal of Artificial Intelligence Research* 57 (2016), 151–185.
[30] Steven R. Ness, Anthony Theocharis, George Tzanetakis, and Luis G. Martins. 2009. Improving automatic music tag annotation using stacked generalization of probabilistic SVM outputs. In *Proceedings of the 17th International Conference on Multimedia (MM 2009)*. Vancouver, CA, 705–708. https://doi.org/10.1145/1631272.1631393
[31] Weike Pan, Erheng Zhong, and Qiang Yang. 2012. Transfer Learning for Text Mining. In *Mining Text Data*, Charu C. Aggarwal and ChengXiang Zhai (Eds.). Springer, Heidelberg, DE, 223–258. https://doi.org/10.1007/978-1-4614-3223-4_7

[32] John C. Platt. 2000. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. In *Advances in Large Margin Classifiers*, Alexander Smola, Peter Bartlett, Bernard Schölkopf, and Dale Schuurmans (Eds.). The MIT Press, Cambridge, MA, 61–74.

[33] Peter Prettenhofer and Benno Stein. 2010. Cross-language text classification using structural correspondence learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*. Uppsala, SE, 1118–1127.

[34] Leonardo Rigutini, Marco Maggini, and Bing Liu. 2005. An EM-based training algorithm for cross-language text categorization. In *Proceedings of the 3rd IEEE/WIC/ACM International Conference on Web Intelligence (WI 2005)*. Compiègne, FR, 529–535. https://doi.org/10.1109/wi.2005.29

[35] Magnus Sahlgren. 2005. An introduction to random indexing. In *Proceedings of the Workshop on Methods and Applications of Semantic Indexing*. Copenhagen, DK.

[36] Magnus Sahlgren and Rickard Cöster. 2004. Using bag-of-concepts to improve the performance of support vector machines in text categorization. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*. Geneva, CH, 487. https://doi.org/10.3115/1220355.1220425

[37] Georgios Sakkis, Ion Androutsopoulos, Georgios Paliouras, Vangelis Karkaletsis, Constantine D. Spyropoulos, and Panagiotis Stamatopoulos. 2001. Stacking classifiers for anti-spam filtering of e-mail. In *Proceedings of the 6th Conference on Empirical Methods in Natural Language Processing (EMNLP 2001)*. Pittsburgh, US, 44—50.

[38] Fabrizio Sebastiani. 2015. An Axiomatically Derived Measure for the Evaluation of Classification Algorithms. In *Proceedings of the 5th ACM International Conference on the Theory of Information Retrieval (ICTIR 2015)*. Northampton, US, 11–20. https://doi.org/10.1145/2808194.2809449

[39] Yangqiu Song, Shyam Upadhyay, Haoruo Peng, and Dan Roth. 2016. Cross-Lingual Dataless Classification for Many Languages.. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence (IJCAI 2016)*. New York, US, 2901–2907.

[40] Philipp Sorg and Philipp Cimiano. 2008. Cross-language Information Retrieval with Explicit Semantic Analysis. In *Working Notes of the 2008 Cross-Language Evaluation Forum (CLEF 2008)*. Aarhus, DE.

[41] Philipp Sorg and Philipp Cimiano. 2012. Exploiting Wikipedia for cross-lingual and multilingual information retrieval. *Data and Knowledge Engineering* 74 (2012), 26–45. https://doi.org/10.1016/j.datak.2012.02.003

[42] Ralf Steinberger, Bruno Pouliquen, Anna Widiger, Camelia Ignat, Tomaz Erjavec, Dan Tufis, and Dániel Varga. 2006. The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. (2006). CoRR abs/cs/0609058.

[43] Kai Ming Ting and Ian H. Witten. 1999. Issues in Stacked Generalization. *Journal of Artificial Intelligence Research* 10 (1999), 271–289.

[44] Grigorios Tsoumakas and Ioannis Katakis. 2007. Multi-Label Classification: An Overview. *International Journal of Data Warehousing and Mining* 3, 3 (2007), 1–13. https://doi.org/10.4018/jdwm.2007070101

[45] Ricardo Vilalta, Christophe Giraud-Carrier, Pavel Brazdil, and Carlos Soares. 2011. Inductive transfer. In *Encyclopedia of Machine Learning*, Claude Sammut and Geoffrey I. Webb (Eds.). Springer, Heidelberg, DE, 545–548.

[46] Alexei Vinokourov, John Shawe-Taylor, and Nello Cristianini. 2002. Inferring a semantic representation of text via cross-language correlation analysis. In *Proceedings of the 16th Annual Conference on Neural Information Processing Systems (NIPS 2002)*. Vancouver, CA, 1473–1480.

[47] Xiaojun Wan. 2009. Co-training for cross-lingual sentiment classification. In *Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing (ACL/IJCNLP 2009)*. Singapore, SN, 235–243. https://doi.org/10.3115/1687878.1687913

[48] David H. Wolpert. 1992. Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259. https://doi.org/10.1016/s0893-6080(05)80023-1

[49] Ting-Fan Wu, Chih-Jen Lin, and Ruby C. Weng. 2004. Probability Estimates for Multi-class Classification by Pairwise Coupling. *Journal of Machine Learning Research* 5 (2004), 975–1005.